© IJEDR 2018 | Volume 6, Issue 2 | ISSN: 2321-9939

Improved fault tolerance architecture for HDFS using Distributed namespaces

[1]Karishma Kharde,[2]Priyanka shelar,[3]Smruti Sakhre
[1],[2],[3]Students
[1]Information Technology

***Abstract*— Interest and Knowledge about the computers and online world is growing along with the need for it. This interest is, no doubt due to continued expansion of internet. This paper is aimed to improve the fault tolerance for HDFS using Distributed Namespace. A center Piece of an HDFS file system, Namespace keeps the directory tree of all files in the file system, also tracks where across the cluster the file data is kept. A client can communicate with the namenode whenever they wish to locate or get a file mapping to the Blocks on Datanodes. A namenode is a Single Point of Failure (SPOF) for the HDFS cluster, when the namenode goes down the file system goes offline there can be an optional name node that can be hosted on a separate machine. Here present multiple namenode with Distributed namespace using Chord Protocol. Multiplenamenodes will be arranged in chord ring to provide fault tolerant architecture in HDFS.**

***IndexTerms*— Chord protocol, HDFS, Namespace, fault tolerant, SPOF**

I. INTRODUCTION

Hadoop is ideal for storing large amounts of data, like terabytes and petabytes and uses a distributed system for data storage called Hadoop Distributed File System(HDFS). There are thousands of server machines in hadoop cluster. Hadoop is an open source, big data storage and processing software framework. Hadoop stores big data in distributed way and processes on large cluster of commodity hardware. HDFS clients perform file system metadata operations through a single server known as the Name-node, and send and retrieve file system data by communicating with a pool of Data-nodes. Data is replicated on each and every data-nodes, hence the loss of a single Data-node should never cause the cluster to fail. But the loss of the Name-node cannot be tolerated. All metadata operations goes to the Name-node, so if the Name-node fails, no clients can read from or write to HDFS. Clients can retrieve individual data blocks present at Data-nodes if the Name-node is fail, but for all purposes, if the Name-node is not available, HDFS is down, then users which depend on HDFS will not be able to function properly. This becomes a Single point-of-failure for an HDFS implementation. A single controller server machine, Namenode is actually essential for the HDFS. This becomes a Single Point of Failure for an HDFS implementation. If the name node goes down then file system goes offline, and the entire system goes into a Halt state. But as soon as it gets back it must respond the client request and datanode manage operations. HDFS includes a secondary namenode, as a replacement for the primary one. In case of Primary node failure, it completely becomes the responsibility of the secondary node to take over the Primary.

Chord's main goal is the location of entities in P2P environments, like documents, files, or any resource that one might want to share in a computer network. It is a distributed lookup protocol which maps a given key onto a node. Data is easily placed in a Chord by associating a key with each resource item. We are also going to usea Metadata Lookup Table for faster lookup.

II. LITERATURE SURVEY

HDFS needs a single controller server machine, the Name-node. This becomes a single point-of-failure for an HDFS implementation. Hence high availability issue of HDFS is addressed by making use of various strategies. This will help to overcome the issue SPOF in Hadoop. And subsequently improved the efficiency of the system. Following are the existing systems used for handling the SPOF :

1. Avatarnode

Aexsiting solution for Name-node failure - To overcome the single point of failure of namenode, almost a few years ago Facebook began work on the Avatarnode. The Avatarnode, which Facebook has contributed back to the community as open source, offers a highly-available Name-node with hot failover and failback. After having many test results, theAvatarnode is now at Facebook running the largest Hadoop Data Warehouse clusters. The Avatarnode is a two-node, highly available Name-node with manualFailover. Avatarnode works by wrapping the existing Name-node code in a Zookeeper layer. The fundamental concepts in Avatarnode are:

i. *There is a Primary and a Standby Avatarnode. One of these can be adopted by the avatarnode.*
ii. *The host name of the current master is kept in Zookeeper*
iii. *A modified Datanode sends block reports to both the Primary and the Standby.*
iv. *An updated HDFS client checks Zookeeper before beginning each transaction, and again at a particular interval of a transaction if one fails. This will allow the write operation to complete even if a Avatarnode fails.*

2.HDFS Federation

HDFS Federation uses various independent namespaces, to scale-up the name services. The name-nodes are federated, that is, the name-nodes are independent and don't require coordination with each other. The data-nodes are used as common storage for blocks by all the name-nodes. Each data-node registers with all the name-nodes in the cluster. Data-nodes sends heartbeats after every some intervals and block reports and also handles commands from the name-nodes. A Block Pool is a set of blocks that belong to a single namespace. Data-nodes store blocks for all the block pools in the cluster. Thus it allow a namespace to generate Block IDs for new blocks without having the coordination among other namespaces. The failure of a name-node does not prevent the data-node from serving other name-nodes in the cluster. A Namespace and its block pool together are called Namespace Volume. It is a self-contained unit of management. The related block pool at the data-nodes is deleted, whenever a namenode/namespace is deleted. Each namespace volume is upgraded as a unit, during cluster upgrade.

III. ALGORITHM

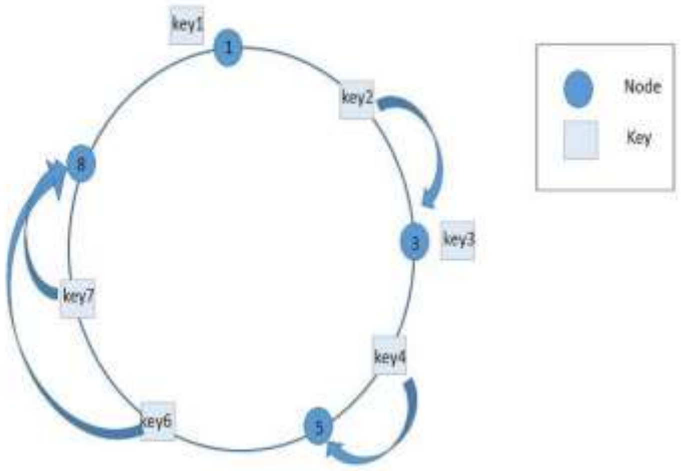
1. Chord Protocol:

Chords main goal is the location of entities in P2P environments, like documents, files,or any resource that one might want to share in a computer network with each resource item.The Function of Chord Protocol is Primitive, for a unique key, it maps a key to a node. Thisnode depending on the application using chord could be in charge for storing a correspondingvalue for a key.It is a distributed lookup protocol which maps a given key onto a node. Data is easilyplaced in a Chord by associating a key.Chord employs consistent hashing to allocate keys tochord nodes.as each node has equal numbers of keys. Consistent Hashing performs load balancing and needs comparatively less reallocations of keys. The figure shows four nodes in the system with identifier 1, 3, 5, 8 respectively. There are 6 keys 1, 2, 3, 4, 6, 7. Key 1 and key
3 is assigned to node 1 and node 3 respectively. As there is no node with identifier 2, key 2 is assigned to node 1 with identifier higher than 2 which is 3.similarly key 4 is assigned to node5 and 6, and key 7 to node 8

2.The Chord Algorithm:

Construction of the Chord ring:

1)The hash function assigns each node and each key an m-bit identifier using SHA1(Secure Hash Standard).
2) m = any number big enough to make collisions improbable
3) Key identifier = SHA-1(key)
4) Node identifier = SHA-1(IP address)
5) Both are uniformly distributed
6) Both exist in the same ID space
7) Identifiers are arranged on a identifier circle modulo 2 = Chord ring
8) A key k is assigned to the node whose identifier is equal to or greater than the keys identifier
9) This node is called successor(k) and is the first node clockwise from k.



3. Simple node localization:

1) ask node n to find the successor of id
2) n.findsuccessor(id)
3) if (id (n; successor))
4) return successor;
5) else
6) forward the query around the circle
7) return successor.findsuccessor(id)

IV. Architecture

The management of ever increasing size of the namespace which holds data related to billions of files comes across as a great challenge. It imposes a threat to high scalability and performance of metadata services. An approach to handle the mentioned threats could be the use of distributed namespaces. This can be done by multiple namenodes instead of the centralized namenode. It involves MLT mechanism for efficient Searching Process.

The Metadata Lookup Table is maintained at the client side. The entries in the MLT consist of the range of hash values and the IP address of Name node responsible for that range. Whenever the client access the file, the file ID is searched into the entries of the MLT and the IP address of the Name node responsible for that file is obtained. Metadata lookup table (MLT) introduces an additional level of indirection between the client and the name node. The Metadata Lookup Table is the structure implemented at client side which helps to directly identify the responsible name node for query.

The proposed architecture addresses the issue of high scalability, Single point of failure(SPOF), High Availability, load balancing in Name-node cluster without compromising the performance. Metadata lookup table(MLT) introduces an additional level of indirection between the client and the Name-node. Chord integration into HDFS Name-node provides a reliable and effective solution to the problem SPOF.

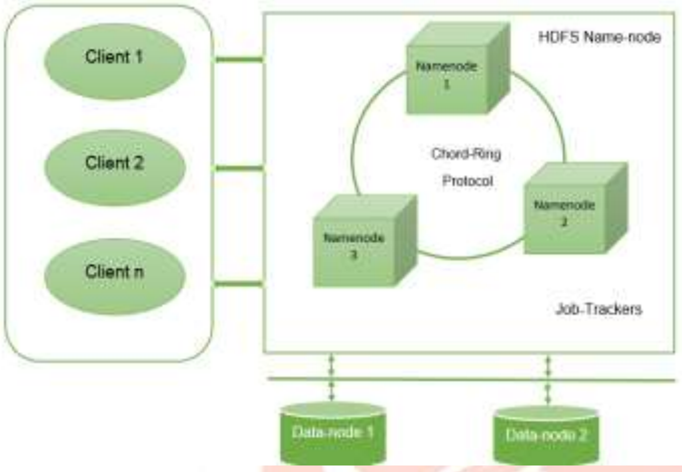


Fig: Proposed Architecture

Whenever client want to perform any operation it will calculate SHA-1 hash of file path.Then it will look up in the MLT for the range in which the file identifier sits, which will ultimately give the IP address of Name node responsible for that request. Now, client can directly contact that name node for operation. Though it looks like multiple Namenodes make HDFS complex, the single point-of- failure HDFS Namenode and its RAM limitation to keep all the files metadata, stored in the Datanodes, required an alternative solution. Chord integration along with HDFS Namenode provides a reliable and efficient solution to the SPOF of namenode.

IV. Conclusion

With the availability of HDFS by enhancing the performance of the system, our contribution focusses on following points. To overcome the problem of Single point of Failure (SPOF) been successfully implemented. And, the dependency of whole system on a single name-node server is decentralized using distributed namespace on multiple name-nodes using a Chord-Ring Protocol. Finally, we implement method Chord-ring protocol to improve the fault tolerance architecture for HDFS that verify the effectiveness of the proposed approach. The system also gives a higher throughput by using fault Tolerance Architecture for HDFS using Distributed namespace. With a rigorous analysis on the characteristics of the operations in HDFS, our uniqueness is to create an adaptive solution to advance the Hadoop system. For further development, some parts of the source code developed to test our idea would be made available under the terms of the GNU general public license (GPL).

REFERENCES

[1] "Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications".
[2] FaragAzzedin, a scalable HDFS architecture," IEEE 2013.
[3] Hasan Tahir, Syed Asim Ali Shah, based Metadata Management for HDFS", 2012 IEEE 14th International Conference on High Performance Computing and Communications.
[4] From Backup to Hot Standby: High Availability for HDFS 2012 31st International Symposium on Reliable Distributed Systems.
[5] Konstas, V. Stathopoulos, and J. M. Jose, "On social networks and collaborative recommendation", in *Proc. 32nd Int. ACM SIGIR Conf. Res. Develop.Inf. Retrieval*, 2009, pp. 195202.
[6] Load rebalancing for Distributed File System in Clouds, IEEE transactions on Parallel and distributed system.
[7] NCluster: Using Multiple Active NamenodestoAchieve High Availability for HDFS 2013 IEEE International Conference on High Performance Computing and Communications 2013 IEEE International Conference on Embedded and Ubiquitous Computing.
[8] An Adaptive Feedback Load Balancing Algorithm in HDFS, 2013 5th International Conference on Intelligent Networking and Collaborative Systems.

[9] Zhipeng Tan, Wei Zhou, Dan Feng, and WenhuaZhang : Adaptive Loading Data Migration in Distributed File Systems, In Proc. IEEE TRANSACTIONS ON MAGNETICS", June, 2013.
[10] Adaptive Loading Data Migration in Distributed File System", IEEE transactions 2013.